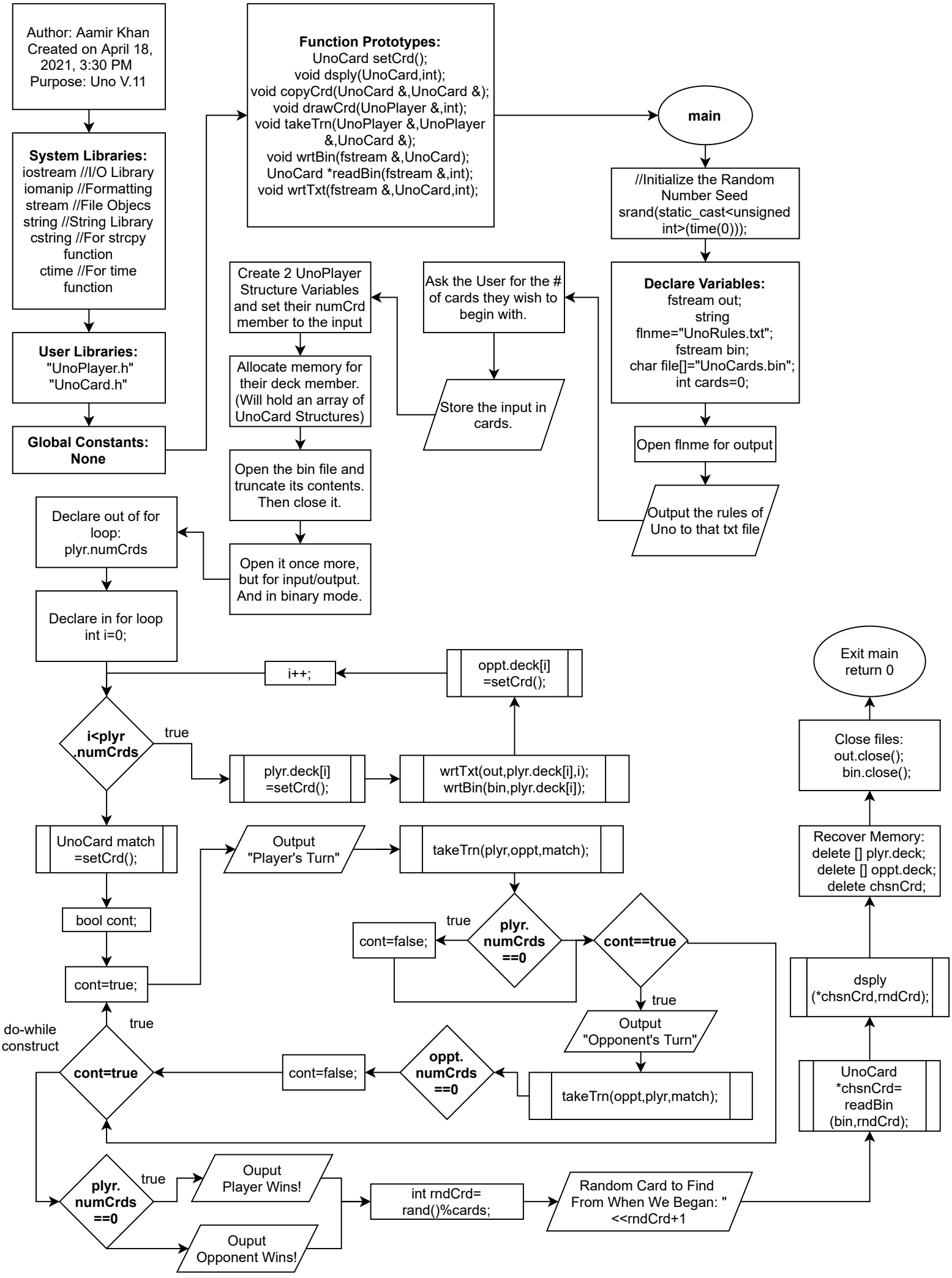
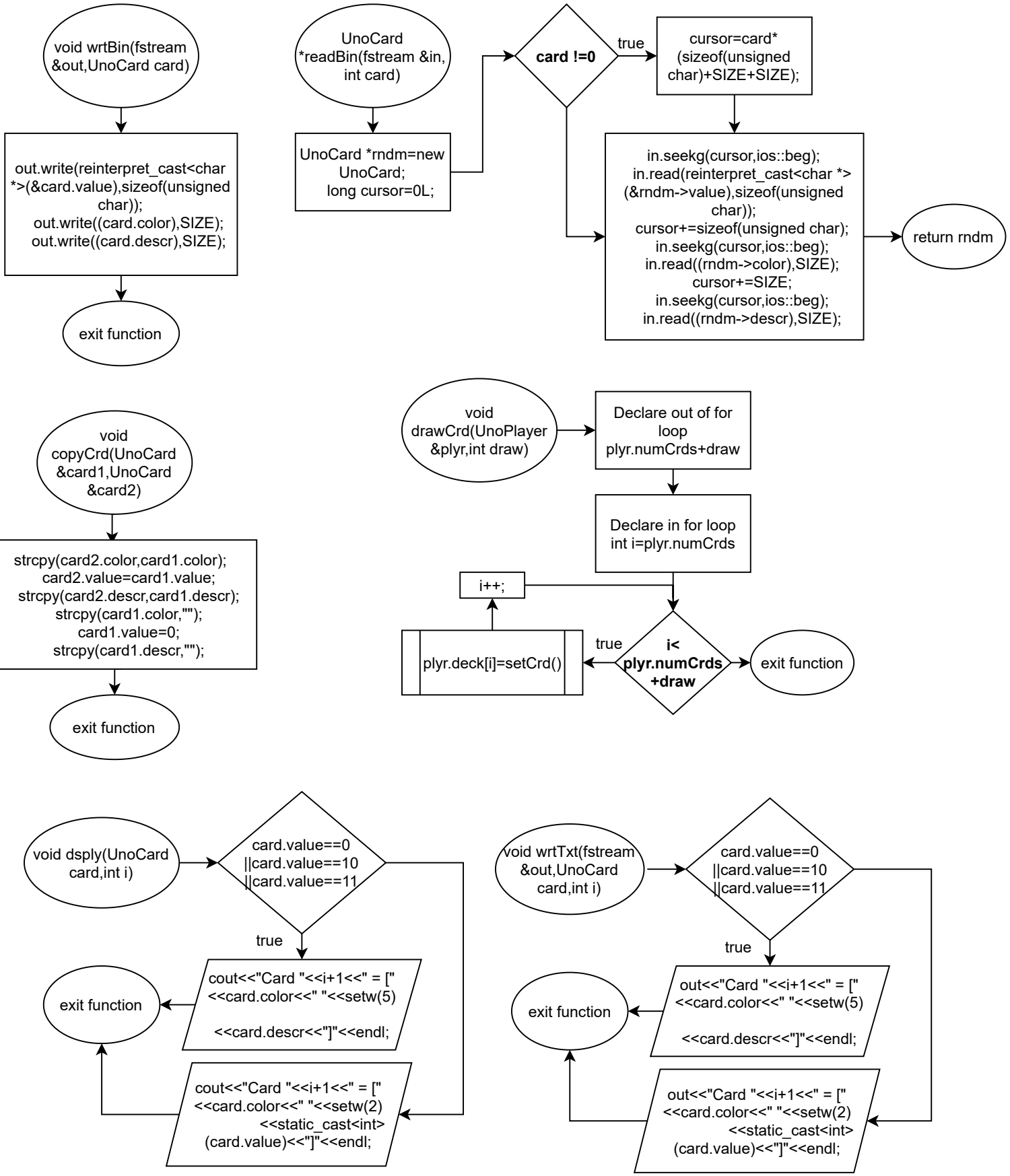
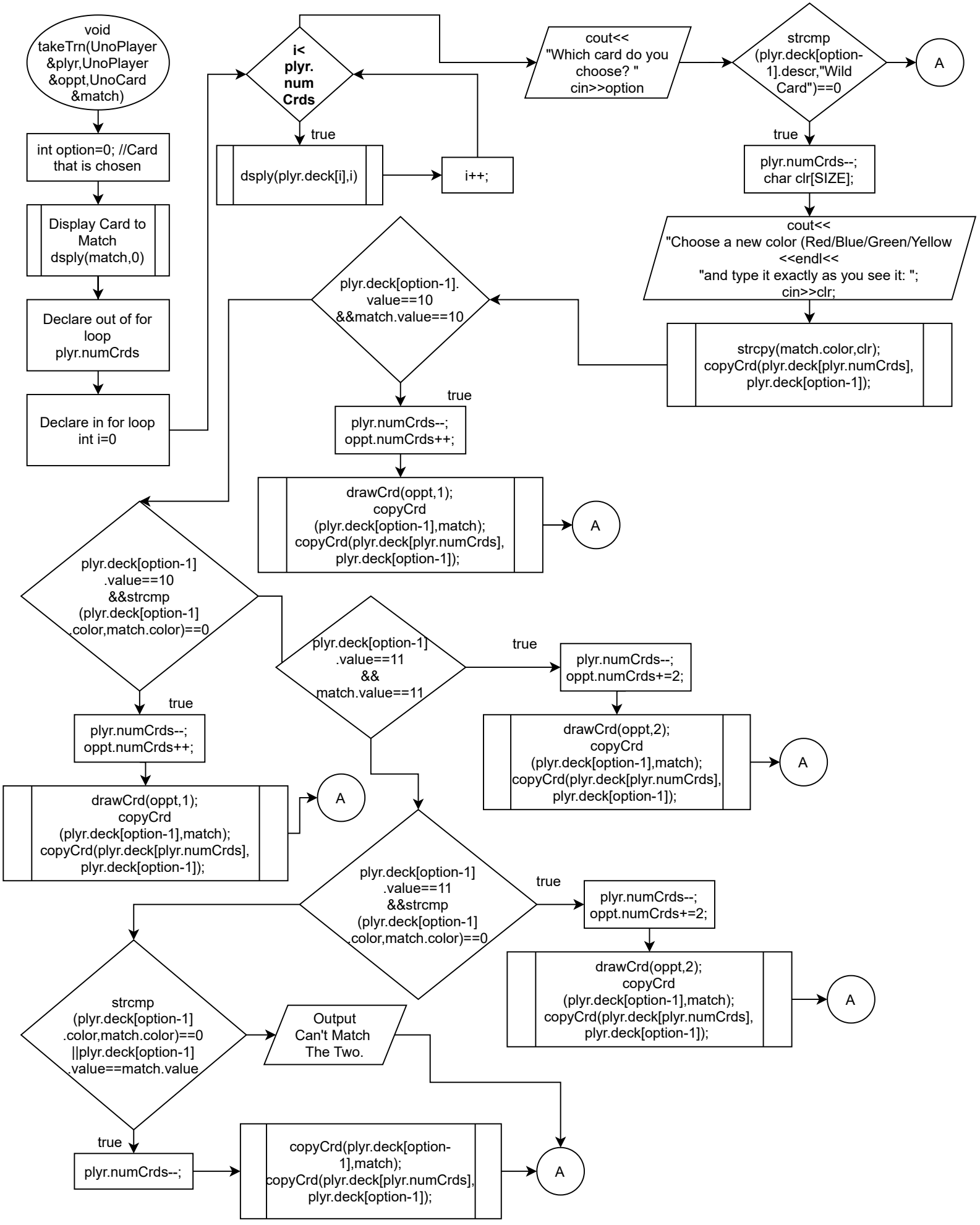
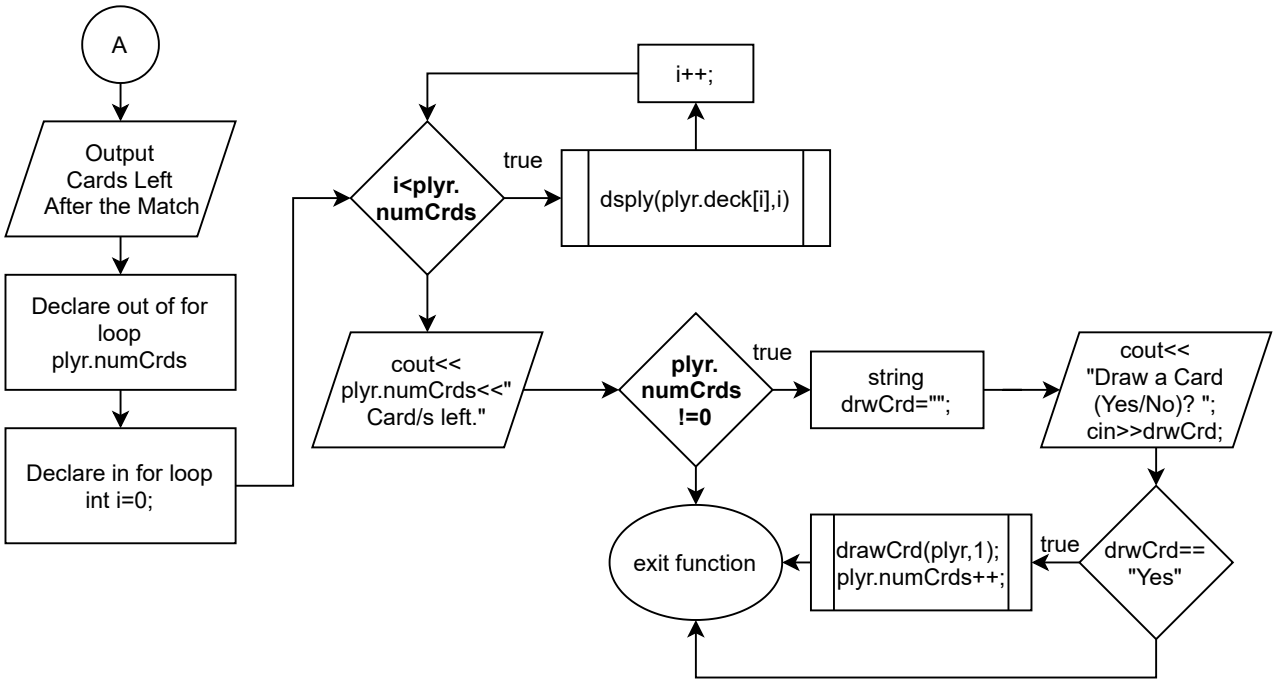

Author: Aamir Khan
Created on April 18, 2021, 3:30 PM
Purpose: Uno V.11
System Libraries:
iostream //I/O Library
iomanip //Formatting stream //File Objecs
string //String Library
cstring //For strcpy function
ctime //For time function
User Libraries:
"UnoPlayer.h"
"UnoCard.h"
Global Constants:
None
Function Prototypes:
UnoCard setCrd();
void dsply(UnoCard,int);
void copyCrd(UnoCard &,UnoCard &);
void drawCrd(UnoPlayer &,int);
void takeTrn(UnoPlayer &,UnoPlayer &,UnoCard &);
void wrtBin(fstream &,UnoCard);
UnoCard *readBin(fstream &,int);
void wrtTxt(fstream &,UnoCard,int);
main
//Initialize the Random Number Seed
srand(static_cast<unsigned int>(time(0)));
Declare Variables:
fstream out;
string flnme="UnoRules.txt";
fstream bin;
char file[]="UnoCards.bin";
int cards=0;
Open flnme for output
Output the rules of Uno to that txt file
Ask the User for the # of cards they wish to begin with.
Store the input in cards.
Create 2 UnoPlayer Structure Variables and set their numCrd member to the input
Allocate memory for their deck member. (Will hold an array of UnoCard Structures)
Open the bin file and truncate its contents. Then close it.
Open it once more, but for input/output. And in binary mode.
Declare out of for loop: plyr.numCrds
Declare in for loop int i=0;
i<plyr .numCrds
true
oppt.deck[i] =setCrd();
i++;
plyr.deck[i] =setCrd();
wrtTxt(out,plyr.deck[i],i); wrtBin(bin,plyr.deck[i]);
UnoCard match =setCrd();
bool cont;
cont=true;
Output "Player's Turn"
takeTrn(plyr,oppt,match);
plyr. numCrds ==0
true
cont=false;
cont==true
true
Output "Opponent's Turn"
takeTrn(oppt,plyr,match);
oppt. numCrds ==0
cont=false;
do-while construct
cont=true
true
plyr. numCrds ==0
true
Ouput Player Wins!
Ouput Opponent Wins!
int rndCrd= rand()%cards;
Random Card to Find From When We Began: " <<rndCrd+1
UnoCard *chsnCrd= readBin (bin,rndCrd);
dsply (*chsnCrd,rndCrd);
Recover Memory: delete [] plyr.deck; delete [] oppt.deck; delete chsnCrd;
Close files: out.close(); bin.close();
Exit main return 0


void wrtBin(fstream &out,UnoCard card)
out.write(reinterpret_cast<char *>(&card.value),sizeof(unsigned char));
out.write((card.color),SIZE);
out.write((card.descr),SIZE);
exit function
UnoCard *readBin(fstream &in, int card)
UnoCard *rndm=new UnoCard;
long cursor=0L;
card !=0
true
cursor=card* (sizeof(unsigned char)+SIZE+SIZE);
in.seekg(cursor,ios::beg);
in.read(reinterpret_cast<char *> (&rndm->value),sizeof(unsigned char));
cursor+=sizeof(unsigned char);
in.seekg(cursor,ios::beg);
in.read((rndm->color),SIZE);
cursor+=SIZE;
in.seekg(cursor,ios::beg);
in.read((rndm->descr),SIZE);
return rndm
void copyCrd(UnoCard &card1,UnoCard &card2)
strcpy(card2.color,card1.color);
card2.value=card1.value;
strcpy(card2.descr,card1.descr);
strcpy(card1.color,"");
card1.value=0;
strcpy(card1.descr,"");
exit function
void drawCrd(UnoPlayer &plyr,int draw)
Declare out of for loop plyr.numCrds+draw
Declare in for loop int i=plyr.numCrds
i++;
plyr.deck[i]=setCrd()
true
i< plyr.numCrds +draw
exit function
void dsply(UnoCard card,int i)
card.value==0 ||card.value==10 ||card.value==11
true
cout<<"Card "<<i+1<<" = [" <<card.color<<" "<<setw(5) <<card.descr<<"]"<<endl;
cout<<"Card "<<i+1<<" = [" <<card.color<<" "<<setw(2) <<static_cast<int> (card.value)<<"]"<<endl;
exit function
void wrtTxt(fstream &out,UnoCard card,int i)
card.value==0 ||card.value==10 ||card.value==11
true
out<<"Card "<<i+1<<" = [" <<card.color<<" "<<setw(5) <<card.descr<<"]"<<endl;
out<<"Card "<<i+1<<" = [" <<card.color<<" "<<setw(2) <<static_cast<int> (card.value)<<"]"<<endl;
exit function


void takeTrn(UnoPlayer &plyr,UnoPlayer &oppt,UnoCard &match)
int option=0; //Card that is chosen
Display Card to Match dsply(match,0)
Declare out of for loop plyr.numCrds
Declare in for loop int i=0
i< plyr. num Crds
true
dsply(plyr.deck[i],i)
i++;
cout<< "Which card do you choose? " cin>>option
strcmp (plyr.deck[option-1].descr,"Wild Card")==0
A
true
plyr.numCrds--; char clr[SIZE];
cout<< "Choose a new color (Red/Blue/Green/Yellow <<endl<< "and type it exactly as you see it: "; cin>>clr;
strcpy(match.color,clr); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
plyr.deck[option-1]. value==10 &&match.value==10
true
plyr.numCrds--; oppt.numCrds++;
drawCrd(oppt,1); copyCrd (plyr.deck[option-1],match); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
A
plyr.deck[option-1] .value==10 &&strcmp (plyr.deck[option-1] .color,match.color)==0
true
plyr.numCrds--; oppt.numCrds++;
drawCrd(oppt,1); copyCrd (plyr.deck[option-1],match); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
A
plyr.deck[option-1] .value==11 && match.value==11
true
plyr.numCrds--; oppt.numCrds+=2;
drawCrd(oppt,2); copyCrd (plyr.deck[option-1],match); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
A
plyr.deck[option-1] .value==11 &&strcmp (plyr.deck[option-1] .color,match.color)==0
true
plyr.numCrds--; oppt.numCrds+=2;
drawCrd(oppt,2); copyCrd (plyr.deck[option-1],match); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
A
strcmp (plyr.deck[option-1] .color,match.color)==0 ||plyr.deck[option-1] .value==match.value
Output Can't Match The Two.
true
plyr.numCrds--;
copyCrd(plyr.deck[option-1],match); copyCrd(plyr.deck[plyr.numCrds], plyr.deck[option-1]);
A


A
Output
Cards Left
After the Match
Declare out of for
loop
plyr.numCrds
Declare in for loop
int i=0;
i<plyr.
numCrds
true
dsply(plyr.deck[i],i)
i++;
cout<<
plyr.numCrds<<"
Card/s left."
plyr.
numCrds
!=0
true
string
drwCrd="";
cout<<
"Draw a Card
(Yes/No)? ";
cin>>drwCrd;
drwCrd==
"Yes"
true
drawCrd(plyr,1);
plyr.numCrds++;
exit function